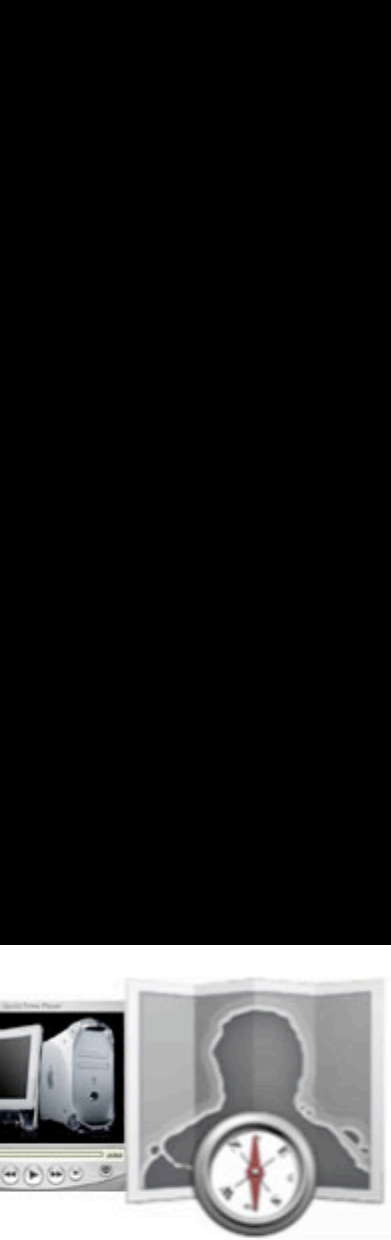


HIToolbox: Introducing HIView

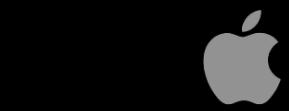
Session 205

HIToolbox: Introducing HIView

Ed Voas
HIToolbox Manager

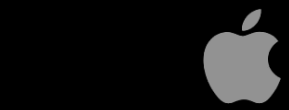
What We Will Learn

- History of controls on Mac OS
- Introducing HIView
    - What it is
    - Advantages
    - How to use it



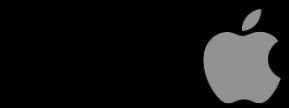
Control Manager

- Introduced in 1984
- Some improvements in System 7
- Lots of new features in Mac OS 8 and beyond
    - Embedding
    - Many new controls
    - Real APIs for structure stuffers



It Ain't Perfect . . .

- 16-bit coordinate space
- Erase-behind behavior
- At times, failure to clip to parent
- Inability to detach/attach subviews easily
- Overlapping views not supported



Introducing HIView

What Is HIView?

- New API to replace Control Manager
- Control Manager is compatibility API
- It is a real view system (no, really)
- This change will not affect you
    - Unless you want to take advantage of it

Advantages

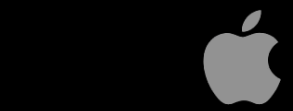
- Consistency and simplicity
    - Reduce and simplify!
- Efficient drawing
- Unification of implementation
    - Everything is a view! (almost)
- Easier to write custom widgets

Still Migrating . . .

- In Jaguar, HIView replacements incomplete
- If not in HIView.h, use Control Manager API
- Over time, HIView.h will subsume Controls.h

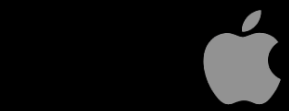
Controls vs. Views

- Controls are Views
- Views are Controls
- The difference is how they act



Features of HIView

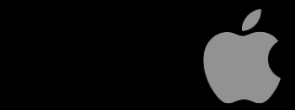
- One-pass, composited draw model
- Modern coordinate system
- Quartz drawing
- Proper Z-ordering
- Ability to attach/detach views at will



Composited Drawing

“Draw or draw not. There is no erase.”

— Anonymous Jedi Master



Composited Drawing

- Predictable drawing order
  - Back to front
  - Respects hierarchy
  - Respects Z-Order
- Eliminates pattern alignment issues

Efficient Drawing

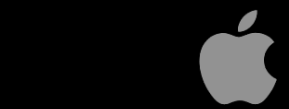
- Goal: draw each opaque pixel once
- Only draw visible area
- How do we know what is visible?
    - Parent bounds
    - Siblings above view in Z-order
    - Opacity of views above you

Invalidation

- Direct drawing is discouraged
- If you want to redraw, you must invalidate
  - HIViewSetNeedsDisplay
  - HIViewSetNeedsDisplayInRegion
- Drawing happens later, at predictable times
  - Right before window flush
  - Window painting

Less Is More

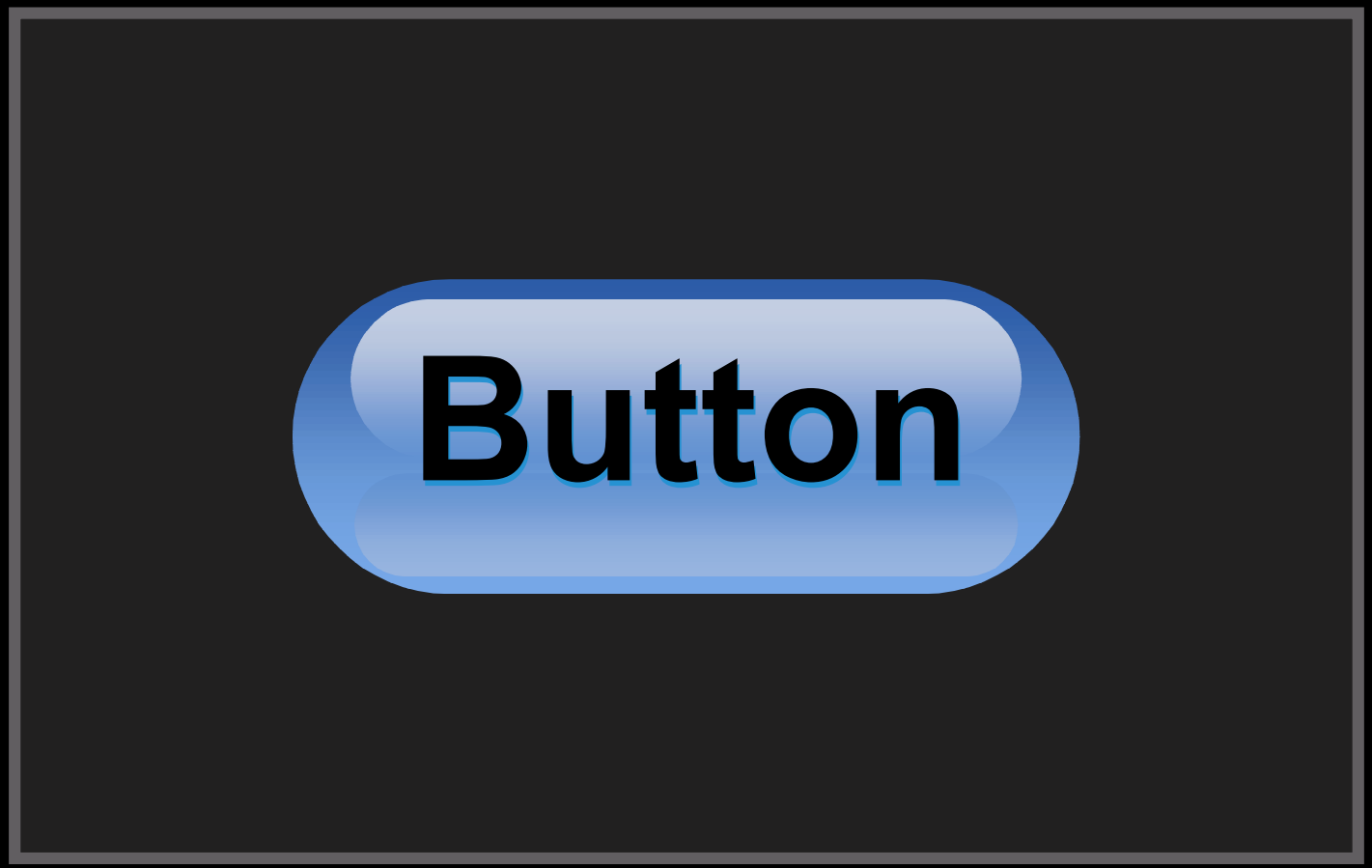
- General Rule: Toolbox never invalidates
  - It is up to the views
- Well OK, sometimes it does
  - Hide, show, move, resize
- View is responsible for invalidation otherwise
  - Activate, enable, value changes, etc.



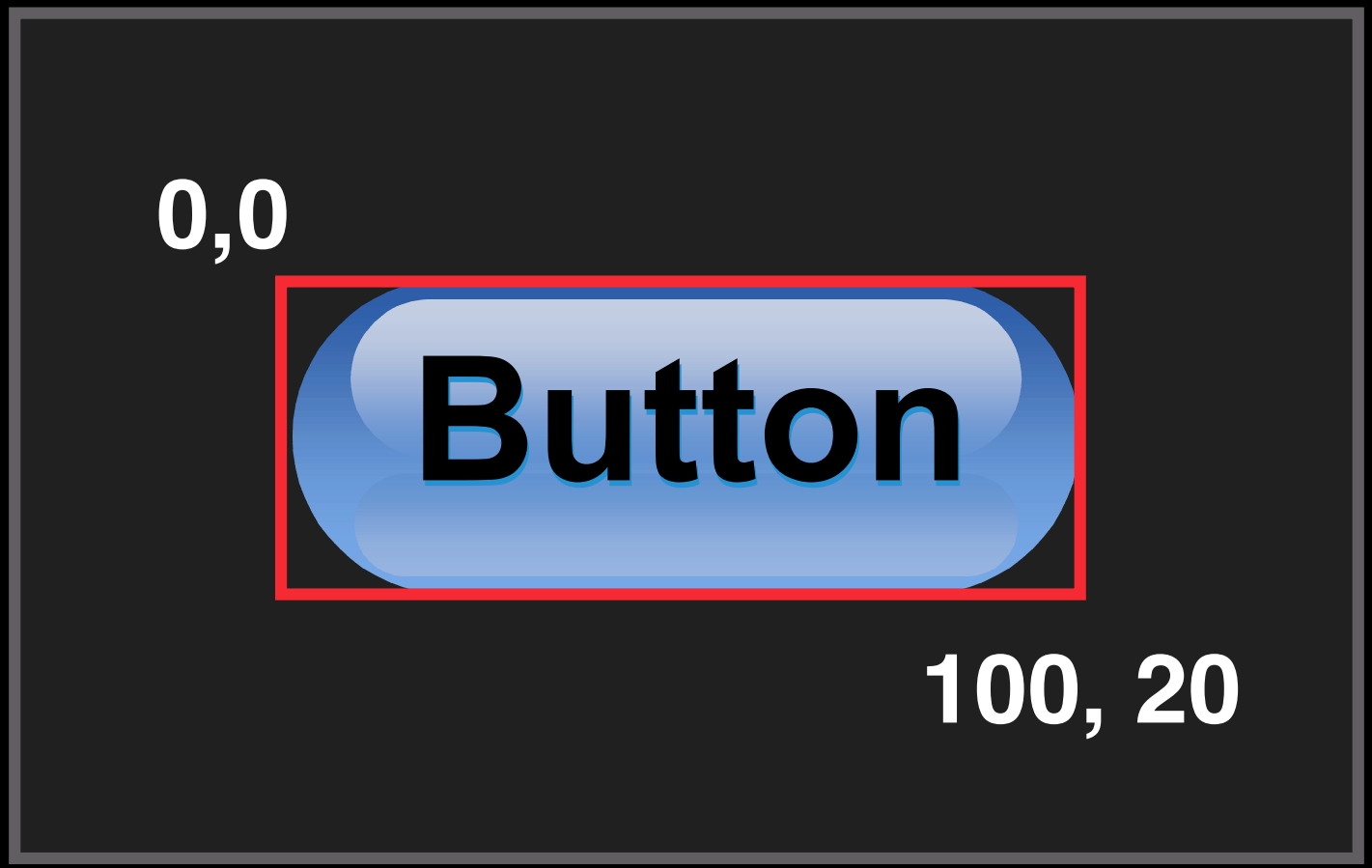
Coordinate Systems

- Two rectangles for every view
  - Frame: Position within parent view
  - Bounds: View's local coordinates
- You move your view based on frame
- You draw and hit test based on bounds

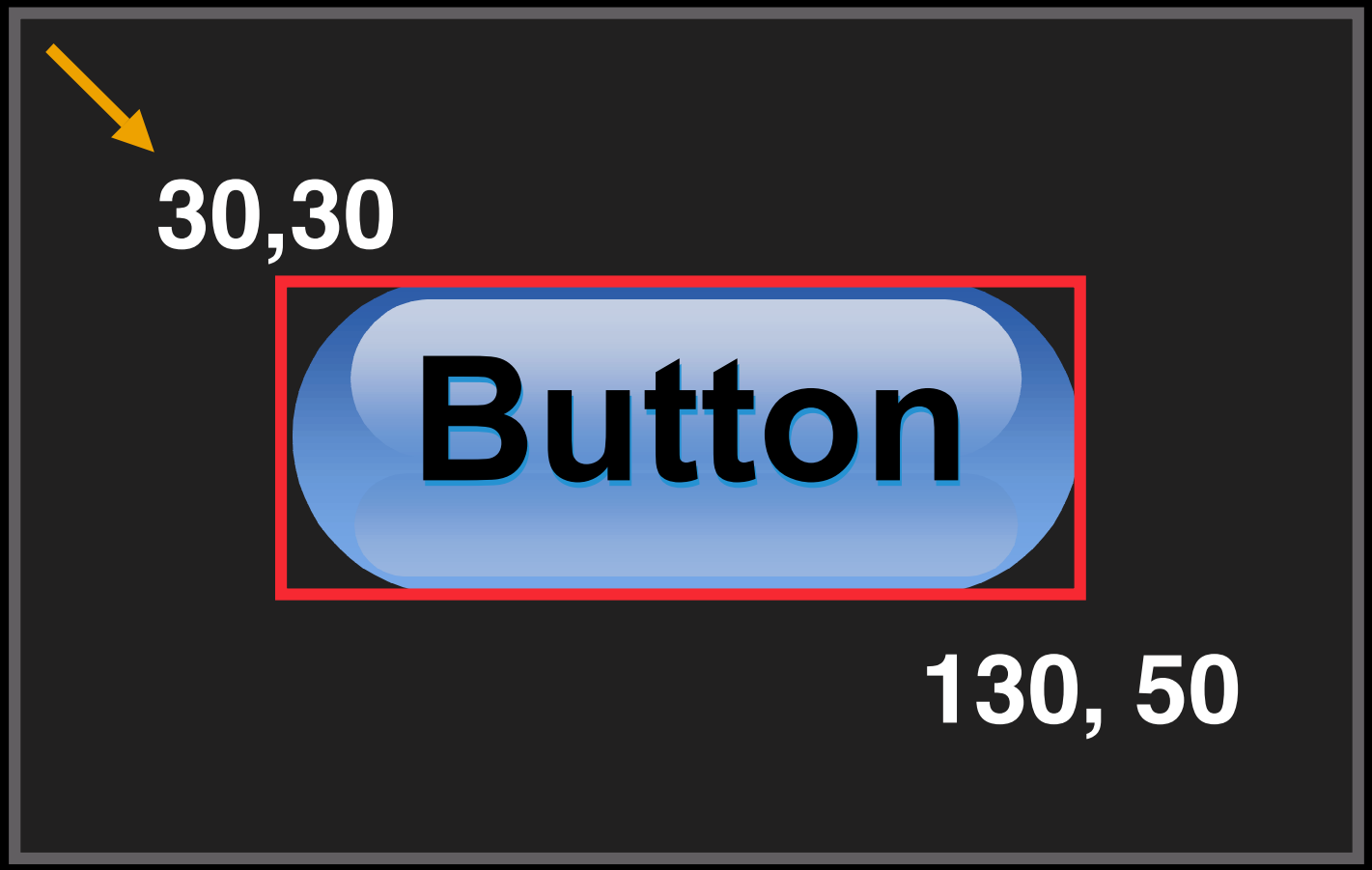
HIView Bounds



HIView Bounds



HIView Frame



Bounds/Frame Advantages

- Consistency
  - These rectangles never change on you
- Frame concept is more efficient
  - Subviews never need to change when parent moves

Positioning Views

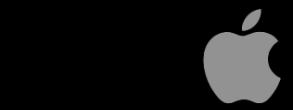
- Was two-stage process
  - Move, resize
- SetControlBounds was a step forward
- Now it is much better
  - HIViewSetFrame
  - HIViewMoveBy
  - HIViewPlaceInSuperviewAt

Converting Coordinates

- Three routines to help you
  - HIViewConvertPoint
  - HIViewConvertRect
  - HIViewConvertRegion
- Converting to/from the NULL view means window relative

New Geometry Types

- Floating-point types to replace QD types
    - HIPoint
    - HIRect
    - HISize
- All HIView APIs are in terms of these new types
- Same as CG types (typedefs)



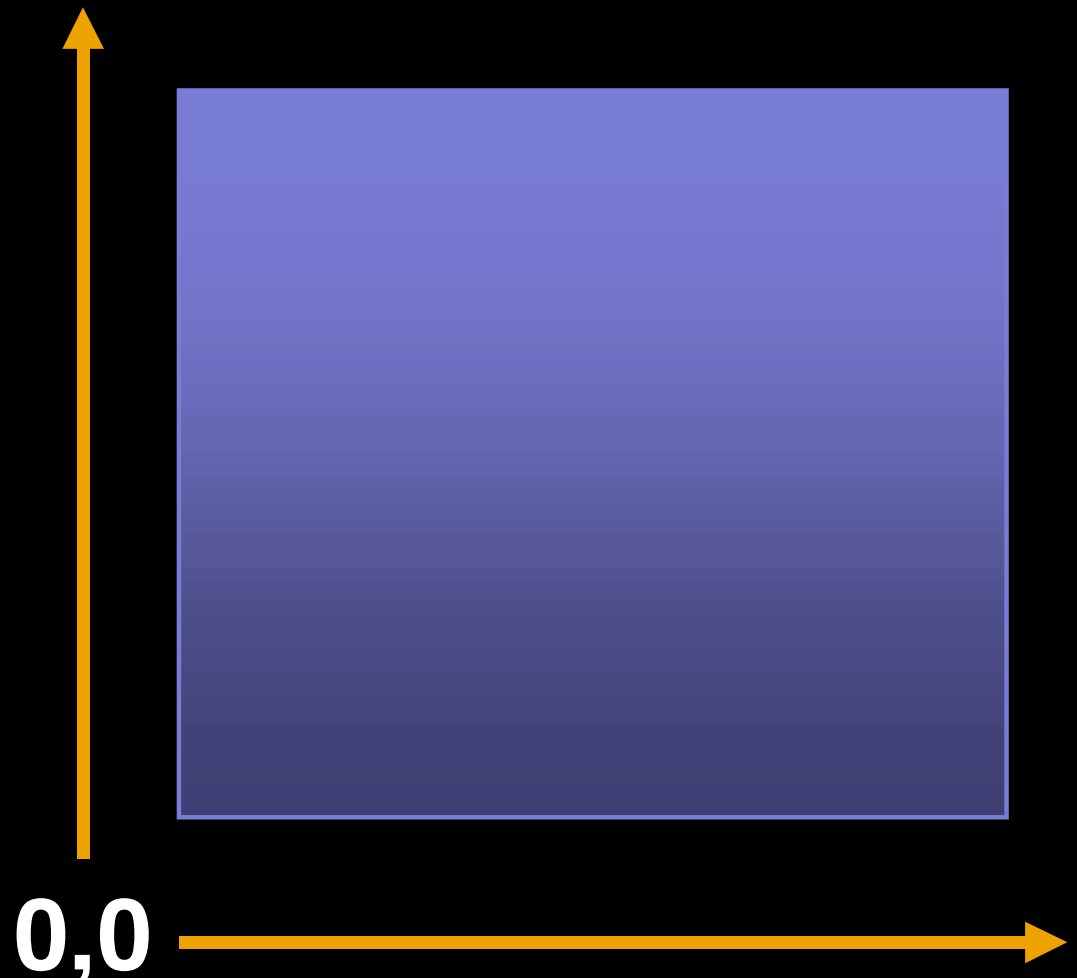
Two Graphics Models

- Quartz
  - Recommended: native model
- QuickDraw
  - Compatibility and quick migration

Quartz

- Native drawing system for Mac OS X
- Allows transparency, anti-aliasing
- Super-cool rad tricks
    - Beziers
    - Scaling
    - Rotating

Quartz Coordinates



HIView Coordinates

Using Quartz

- Context is transformed— 0, 0 at top left
  - Windowing system is top left
  - Your existing code is top left
  - Do not need to deal with window resize issues
- Requires some new APIs to draw images
  - HIViewDrawCGImage

QuickDraw Drawing

- For easy transition, QD possible
  - Your view must advertise its need for QD
  - Performance tax—need to sync

Off-screen Imaging

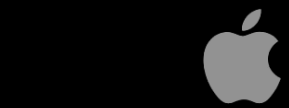
- Drawing controls into off screens was difficult
- Now it is easy!
    - HIViewCreateOffscreenImage
- Yields CGImageRef
- Will be able to use for drag images, etc.

Demo

Enabling HIView

Enabling Compositing

- Per-window
- Must use CreateNewWindow
- kWindowCompositingAttribute
    - Specifiable only at window creation time
- Standard handler is highly recommended



Windows With Compositing

- Operates completely in terms of views
- Painting merely paints the root view
  - Paint procs are not called
  - Content is not erased as usual
- When window resized, root view is resized
  - Layout of children propagates from there
- Listen to kEventControlParentBoundsChanged

Windows With Compositing

- When window is hidden, so is root view
  - Prevents unnecessary drawing
- When window is shown, view is shown/drawn
  - Allows content to be prerendered
- Do not need to listen to kEventWindowDrawContent

The View Hierarchy

Start at the Top

- With compositing on, hierarchy is deeper
- Root view is the window frame
    - Window widgets are views themselves
    - Contains a content view
- From root you can find any standard subview
    - HIViewFindByID

Effect on CreateRootControl

- In compositing mode, CreateRootControl will err
  - errRootAlreadyExists
- GetRootControl will yield the content root
  - Maintains compatible behavior
- HIViewGetRoot will yield the true root
- Use HIViewFindByID
  - kHIViewWindowContentID

Embedding

- Add new subviews . . .
  - HIViewAddSubview
- Remove subviews . . .
  - HIViewRemoveFromSuperview
- New controls we introduce will not have an owning window parameter

Z-Order Fun

- We allow Z-order iteration
  - HIViewGetFirstSubView
  - HIViewGetLastSubView
  - HIViewGetNextView
  - HIViewGetPreviousView
- And moving in the Z-order
  - HIViewSetZOrder

Mouse Event Handling

Mouse Handling: The Past

- Get event
- Call FindWindow
- If the result is in Content, call FindControl
    - Call TrackControl/HandleControlClick
- If result in window widget, call TrackBox
    - If successful, do whatever is expected

Mouse Handling: Now

- Get event
- Determine window from event parameters
- Find out which view of window to give it to
- Give the event to that view
- You do not treat window widgets differently!

New Event Parameters

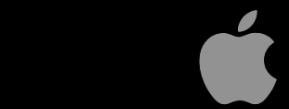
- kEventParamWindowRef
    - What window the mouse event is for
- kEventParamWindowMouseLocation
    - Window structure-relative mouse location

Mouse Event APIs

- HIViewGetViewForMouseEvent
  - Tells you what view should receive it
  - Allows parent views to intercept clicks in their children
- HIViewClick
  - Sends a mouse down event to a view
  - Can turn into a context click or normal one

Mouse Event APIs

- HIViewGetSubViewHit
  - Raw subview hit detection
  - Generally not used
- HIViewSimulateClick
  - Great for making custom views accessible



Handling Mouse Events

- Could not get easier

```
rootView = HIViewGetRoot( window );
HIViewGetViewForMouseEvent( root, event, &target );
result = HIViewClick( target, event );
```

- Well, I lied . . . use standard window handler!

It Is Enabled—Now What?

- Eliminate calls to DrawControls, etc.
- Evaluate usage of Get/SetControlBounds
    - HIViewGet/SetFrame
    - HIViewGetBounds
- Eliminate any erase-behind you might be doing

Custom Content

- Must be wrapped into an HIView
- Not as hard as it sounds in most cases
- Required to ensure consistent behavior

Creating Custom Views

- Custom views need to subclass HIView

```
#define kMyPhatClassID
            CFSTR( “com.phatclass.superwidget” )

HIObjectRegisterSubclass(
    kMyPhatClassID,
    kHIViewClassID,
    0, // no options
    MyPhatEventHandler,
    GetEventTypeCount( kEvents ),
    kEvents,
    NULL, // no handler data
    &classRef );
```

Creating Custom Views

- Next, you need to instantiate your view

```
HIObjectCreate(
    kMyPhatClassID,
    NULL, // or  initialization event as needed
    &objectRef );
```

Typical Events to Implement

- Hit Testing
- Drawing
- Region Calculation
- Drag and Drop

Simple Hit Testing Example

```
HIViewPartCode
MyObject::GetPartHit( const HIPoint& inPoint )
{
    HIRect                  bounds;
    HIViewPartCode          part = kControlNoPart;

    HIViewGetBounds( GetViewRef(), &bounds );

    if ( CGRectContainsPoint( bounds, inPoint )
        part = kMyControlPart;

    return part;
}
```

The Draw Method

- Carbon Event: kEventControlDraw
- Two parameters
    - Draw region: how much to draw
    - Context: where to draw

Simple Drawing Example

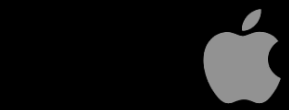
```
void
MyObject::Draw(    RgnHandle          inLimitRgn,
                   CGContextRef       inContext )
{
    HIRect                    bounds;

    HIViewGetBounds( GetViewRef(), &bounds );

    CGContextSetRGBFillColor( inContext, 1, 0, 0, 1 );
    CGContextStrokeRect( inContext, bounds );
}
```

Calculating Regions

- Carbon Event: kEventControlGetRegion
- When the Toolbox needs certain regions, it asks your view
- Two most important regions
    - Structure region
    - Opaque region

Structure Region

- This can extend outside your bounds
    - Focus rings
- If you do not respond, your bounds is your structure
- If you ever need to reshape yourself
    - HIViewReshapeStructure
    - Recomputes your structure and invalidates



Opaque Region

- Helps determine what is visible below you
- Optimizes drawing
- Used to determine window's opaque region
- If you do not respond when asked for this region, you are completely transparent

Drag and Drop

- Supported through several new Carbon Events
  - kEventControlDragEnter
  - kEventControlDragWithin
  - kEventControlDragLeave
  - kEventControlDragReceive
- You must turn on drag support for the window
  - SetAutomaticControlDragTracking-EnabledForWindow (whew!)

Drag and Drop

- If you do not respond to drag enter . . .
  - No within, leave, or receive events for you!
  - For efficiency
- If you may want the drag, return noErr from your handler
  - But do not do so if you know you do not

Drag and Drop

- Innermost ‘focused’ view gets ‘within’ events
    - Containing views do not
- Also is target of the drop

Demo

Summary

- HIView is a huge step forward for Carbon!
  - It frees us from the past
- Our future is clear
  - Carbon Events, HIObject, HIView
- You are urged to at least start working with the WWDC seed
  - Feedback is critical to its success!

Roadmap

206 HIToolbox: New Controls and Services: Learn about the new tools at your disposal	Hall 2 **Wed., 2:00pm**
207 Improving Performance With Carbon Events: How to make the most of Carbon Events	Hall 2 **Wed., 3:30pm**
FF005 HIToolbox: Come tell us what you think	Room J1 **Thurs., 10:30am**

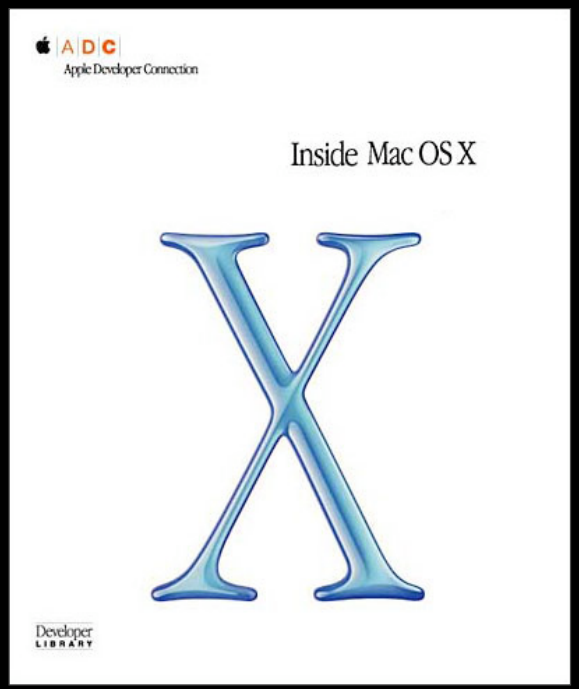
Who to Contact

Xavier Legros
Mac OS X Evangelist
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html

Documentation

HIView

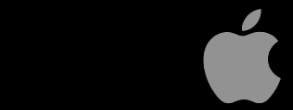


- HIView Reference (Prelim)
- HIObject Reference (Prelim)
- HIToolbar Reference (Prelim)

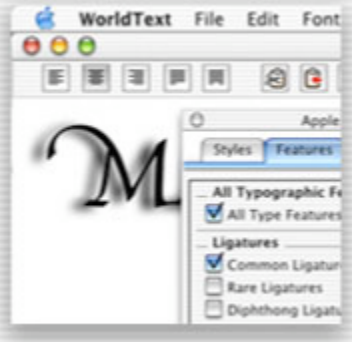
ADC Member Site > Download Software > “Jaguar” Mac OS X

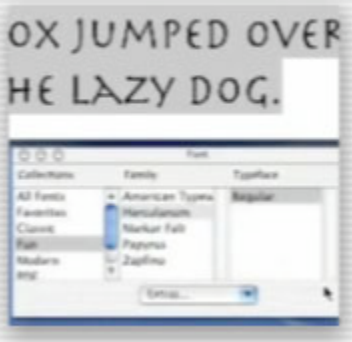
For More Information

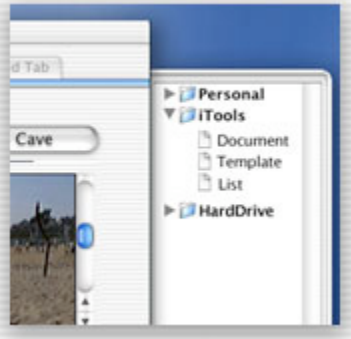
- O'Reilly "Learning Carbon"
- Carbon Developer Documentation
  http://developer.apple.com/techpubs/macosx/macosx.html



Q&A







Xavier Legros
Mac OS X Evangelist
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002